

REMARKS

The claims are claims 1 to 7.

The application has been amended at many locations to correct minor errors and to present uniform language throughout. The amendments include correction of those errors noted by the Examiner. These amendments include an update of the status of the co pending applications cited on page 1. These amendments include renumbering some reference numbers in Figure 4 that were inadvertently used for differing structures.

New formal drawing including the corrections required by the Examiner are attached. The proposed drawing correction corrects the spelling of "SYSCLK" in Figure 4 to match the text at page 11, line 10. The proposed drawing correction changes some reference numbers of Figure 4 to avoid using these reference numbers for differing structures.

Claim 1 is amended to clarify the subject matter and further distinguish over the reference. Claims 4 to 7 are added. Claims 4 and 5 recite subject matter taught in the application at page 29, lines 25 to 32. Claims 6 and 7 recite subject matter taught in the application at page 16, lines 8 to 19 and page 16, line 31 to page 7, line 4 and illustrated in Figure 6.

The Applicant respectfully submits that the specification of this application is proper under 35 U.S.C. 112. The application has been amended to include the serial numbers of the co-pending applications cited on page 1, which were not available when this application was filed. The patent numbers of applications that have been granted as of the date of this response have been included. Accordingly, the Applicant respectfully submits this is proper incorporation by reference.

The Applicant respectfully submits that this application is proper under 35 U.S.C. 112 even without the incorporation of the

listed co-pending applications. In claim 1: the limitation "detecting a first debug event during normal program execution" is known in the art; the limitation "suspending normal program execution while permitting at least one type interrupt service routine executed in response to a corresponding interrupt" is described in the application at page 5, lines 5 to 7, and page 13, line 27 to page 14, line 6; the limitation of "incrementing a debug frame counter upon each of the at least one type interrupt received while suspending normal program execution" is described in the application at page 5, lines 6 to 9 and page 32, lines 21 to 24; the limitation of "decrementing the debug frame counter upon each return from interrupt received while suspending normal program execution" is described in the application at page 5, line 8, page 15, lines 1 and 2, and page 17, lines 3 and 4; the limitation of "detecting at least one second debug event during an interrupt service routine executing while suspending normal program execution" is described in the application at page 5, lines 9 to 11, and page 1, lines 13 and 14; the limitation of "upon detection of the second debug event suspending program execution of the interrupt service routine while permitting execution of other interrupt service routines in response to corresponding interrupts" is described in the application at page 5, lines 9 and 10, page 22, lines 15 to 18; and the limitation of "storing the count of said debug frame counter upon each second debug event" is described in the application at page 5, line 11, page 29, lines 21 to 24. The limitations of claim 2 are described in the application at page 5, lines 14 to 18, page 31, lines 6 to 9, and page 32, lines 7 to 9. The limitations of claim 3 are described in the application at page 5, lines 18 to 20, and page 32, line 25 to page 33, line 3.

Claims 1 to 3 were rejected under 35 U.S.C. 112 as containing subject matter which was not described in the specification. The OFFICE ACTION cites AFEN of page 18 as making it unclear to one

skilled in the art how to generate a debug suspend request without undue experiment. The OFFICE ACTION further states that essential matters are improperly incorporated by reference.

The Applicant respectfully submits that the original application is sufficient as required by 35 U.S.C. 112. The complained citation to AFEN has been deleted. This reference was to an example of detecting a debug event. This process is known in the art and the previous reference to AFEN was merely an example of how to detect a debug event. The Applicant respectfully submits that the application teaches how address comparison unit 310, data comparison unit 320 or external comparison unit 330 can trigger debug events without the reference to AFEN.

As noted above, the Applicant believes that the amendments to the application now provide proper incorporation by reference and that the application as filed teaches the claimed subject matter in the manner required by 35 U.S.C. 112 even without the incorporation by reference.

Claims 1 to 3 were rejected under 35 U.S.C. 102(e) as anticipated by Brannick et al., U.S. Patent 6,289,300.

Claim 1 recites subject matter not anticipated by Brannick et al. Claim 1 recites "upon detection of the first debug event suspending normal program execution while permitting at least one type interrupt service routine executed in response to a corresponding interrupt." Brannick et al states at column 2, line 66 to column 3, line 2 (cited in the OFFICE ACTION):

"Preferably the receipt of an instruction to commence emulation causes a high level non maskable interrupt to be issued to the data processing core of the data processor."

Brannick et al further states at column 5, lines 23 to 28:

"The interrupt controller 30 is arranged to buffer incoming interrupts while the programme address controller is responsive to the emulation request controller 32 in order to ensure that interrupt requests are properly serviced upon return from the emulation mode."

Brannick et al further states at column 6, lines 3 and 4:

"This enables the emulator to halt internal timers and interrupts when executing an emulation function."

These portions of Brannick et al clearly differ from the claimed subject matter. First, the portion cited by the Examiner states that emulation is begun by a high level non maskable interrupt. Second, the quoted portions of columns 5 and 6 indicate that interrupts are not enabled in debug suspend mode. Thus the interrupts noted by the Examiner are not received and serviced while normal program execution is suspended by debug as recited in claim 1. Accordingly, claim 1 is allowable over Brannick et al.

Claim 1 recites further subject matter not anticipated by Brannick et al. Claim 1 recites "incrementing a debug frame counter upon each of the at least one type interrupt received while suspending normal program execution and "decrementing the debug frame counter upon each return from interrupt received while suspending normal program execution." The OFFICE ACTION cites the auxiliary programme address counter disclosed at column 3, lines 8 to 14 of Brannick et al as anticipating the claimed debug frame counter. The portion of Brannick et al cited by the Examiner disclosing using the auxiliary programme address counter rather than the programme address counter used by executable code to preserve the state of the data processor before emulation. Brannick et al states at column 4, lines 4 to 19 states:

"FIG. 4 schematically illustrates the internal arrangement of a data processor, generally illustrated as 20, whereby a

programme address controller 22 holds the address of the next memory location to be read from. This location normally points to a boot strap memory 24 which contains executable code to be used during power up sequences or for data exchange routines, or to a user programme 26 which may be stored in an internal user code memory 28 or in external memory (not shown). The programme address controller 22 is responsive to an interrupt request handle 30 which, as is well known in the art, allows normal execution of a programme to be interrupted in order to respond in a predetermined way to specific events. The data processor is arranged to execute an interrupt routine in response to an interrupt request and then to return to the user code when the interrupt routine has been completed."

Since Brannick et al teaches that the auxiliary programme address counter is used as a substitute for the programme address counter, this structure operates in the same fashion. Brannick et al fails to teach that the auxiliary programme address counter is incremented upon an interrupt. Instead, Brannick et al teaches a data processor "arranged to execute an interrupt routine in response to an interrupt request." As known in the art, this involves placing the address of the interrupt service routine corresponding to the received interrupt into the programme address counter. This address then becomes "the next memory location to be read from," the location of code for response to the interrupt. Brannick et al fails to teach that the auxiliary programme counter is decremented upon return from an interrupt. Instead, Brannick et al teaches a data processor arranged "to return to the user code when the interrupt routine has been completed." As known in the art, this involves placing the address of the prior location of the user code into the programme address counter. Thus the data processor proceeds with the user code at the location before the interrupt. Accordingly, claim 1 is allowable over Brannick et al.

Claim 1 recites further subject matter not anticipated by Brannick et al. Claim 1 recites "storing the count of said debug frame counter upon each second debug event." The OFFICE ACTION

cites the auxiliary stack disclosed in Brannick et al at column 3, lines 8 to 14 as anticipating the claimed storage. As noted above, Brannick et al teaches the auxiliary stack is used during emulation to leave the stack used by the data processor user unaltered. As known in the art, a stack is a general purpose data storage structure. Brannick et al fails to teach storage of the debug frame count in this auxiliary stack. The disclosure of a general purpose data storage structure does not anticipate all data that may be stored there. Accordingly, claim 1 is allowable over Brannick et al.

Claim 2 recites subject matter not anticipated by Brannick et al. Claim 2 recites "a plurality of debug event decoders" and "said step of storing the count of said debug frame counter occurs at said second one of the plurality of debug event detectors." The OFFICE ACTION cites Brannick et al column 2, lines 61 to 66 as anticipating detecting the first debug event. This portion of Brannick et al states:

"The receipt of the instruction to start emulation [sic emulation], either by a signal on the emulator control pin or via a software instruction within the software being debugged, forces the data processing core of the data processor to suspend execution of the user's programming code and to execute instructions from the emulation instruction code."

This teaches two sources of the instruction to start emulation: 1) a signal on an emulation pin; and 2) a software instruction. Brannick et al states at column 5, lines 44 to 47 (cited in the OFFICE ACTION as anticipating detecting the second debug event):

"Preferably, the emulation controller 32 is responsive to a break point instruction 43 whose occurrence in the user code causes the emulation controller 32 to issue an emulation request."

The Applicant submits that the "break point instruction" of column 5 of Brannick et al is the "software instruction within the software being debugged" of column 3 of Brannick et al. Figures 4 and 5 of Brannick et al illustrate inputs to emulation controller 32 from EA pin 40 and breakpoints 43. The OFFICE ACTION states that the auxiliary stack disclosed at column 3, lines 8 to 14 of Brannick et al anticipates the claimed step of "storing the count of said debug frame counter occurs at said second one of the plurality of debug event detectors." The OFFICE ACTION points out no connection between emulation controller 32 and the auxiliary stack and Brannick et al disclosed no such connection. Thus Brannick et al fails to disclose storing the debug frame count at the second debug event detector as recited in claim 2. Accordingly, claim 2 is allowable over Brannick et al.

Claim 3 recites subject matter not anticipated by Brannick et al. Claim 3 recites "determining an order of interrupts triggering second debug events by reading said stored count of said debug frame counter from each of said debug event detectors." The OFFICE ACTION cites the auxiliary stack disclosed at column 3, lines 8 to 14 of Brannick et al as anticipating this subject matter. However, Brannick et al fails teach the claimed "determining an order of interrupts." Note that the auxiliary stack of Brannick et al is a general purpose data storage structure. However, Brannick et al fails to teach use of data in the manner recited in claim 3. Accordingly, claim 3 is allowable over Brannick et al.


New claims 4 to 7 recite subject matter neither anticipated by nor made obvious by Brannick et al. Claim 4 recites limiting each debug event detector to triggering a single debug event before being cleared. This subject matter is not taught in Brannick et al. Claim 5 recites this limiting includes "prohibiting triggering a debug event if the stored count of the debug frame counter is nonzero." Brannick et al includes no teaching of this limitation.

Claims 6 and 7 recite conditions under which the debug frame count is reset. These conditions are not taught in Brannick et al.

The Applicant respectfully submits that all the present claims are allowable for the reasons set forth above. Therefore early reconsideration and advance to issue are respectfully requested.

If the Examiner has any questions or other correspondence regarding this application, Applicants request that the Examiner contact Applicants' attorney at the below listed telephone number and address to facilitate prosecution.

Texas Instruments Incorporated
P.O. Box 655474 M/S 3999
Dallas, Texas 75265
(972) 917-5290
Fax: (972) 917-4418

Respectfully submitted,

Robert D. Marshall, Jr.
Reg. No. 28,527